VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Získávání EPG informací z DVB-S/S2

Getting EPG Information from the DVB-S/S2

2015 Lukáš Richter

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Lukáš Richter**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Získávání EPG informací z DVB-S/S2
Getting EPG Information From the DVB-S/S2

Zásady pro vypracování:

Navrhněte a realizujte systém, který bude prostřednictvím DVB-S/S2 karty získávat EPG informace vysílané na jednotlivých transpondérech družice. Získaná data ukládejte ve formátu XMLTV tak, aby informace o programech byly uloženy v samostatných souborech. Implementujte možnost získávání dat z více družic prostřednictvím přepínání DiSEqC. Vyvíjené programy budou určeny pro OS Linux.

1. Stručně popište systém EPG a jeho distribuci v systému DVB.
2. Popište API využívané v OS Linux pro řízení DVB karet.
3. Naprogramujte systém pro získávání EPG informací, tak aby mohl běžet jako trvalá služba.
4. Vytvořte jednoduchý webový interface pro prezentaci EPG dat.
5. Systém otestujte.

Seznam doporučené odborné literatury:

[1] ETSI EN 300 468. Digital Video Broadcasting (DVB) : Specification for Service Information (SI) in DVB systems. Sophia Antipolis : ETSI, 2010. 137 s. Dostupné z WWW: <http://www.etsi.org/deliver/etsi_en/300400_300499/300468/01.11.01_60/en_300468v011101p.pdf>.

[2] LEGÍŇ, Martin. Televizní technika DVB-T. 1. české. [s.l.] : [s.n.], 2006. 288 s. ISBN 80-7300-204-3.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. David Seidl, Ph.D.**

Datum zadání: 01.09.2014

Datum odevzdání: 07.05.2015

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA * Fakulta elektrotechniky a informatiky * OSTRAVA *

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 *Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava.*

V Ostravě 6. května 2015 .....

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 6. května 2015 .....

Rád bych na tomto místě poděkoval Ing. Davidu Seidlovi, Ph.D. za odborné vedení během mé práce.

Abstrakt

Tato práce se zabývá vývojem a implementací aplikace pro získávání EPG dat z DVB-S/S2 systému. Teoretická část je zaměřena zejména na distribuci servisních informací v DVB systému. Praktická část řeší implementaci aplikace v programovacím jazyku C++. Kromě této aplikace bylo naprogramováno jednoduché webové rozhraní pro zobrazení získaných EPG dat.

Klíčová slova: DVB, EPG, C++

Abstract

This thesis deals with the development and implementation of applications for getting EPG data from DVB-S/S2 system. The theoretical part is focused on the distribution of service information in DVB systems. The practical part describes implementation of application in the C++ programming language. In addition to this application was programmed simple web interface to display obtained EPG data.

Keywords: DVB, EPG, C++

Seznam použitých zkratek a symbolů

ATSC	–	Advanced Television Systems Committee
BAT	–	Bouquet Association Table
CAT	–	Conditional Access Table
CSS	–	Cascading Style Sheets
DIT	–	Discontinuity Information Table
DVB	–	Digital Video Broadcasting
DVB-C	–	Digital Video Broadcasting - Cable
DVB-H	–	Digital Video Broadcasting - Terrestrial
DVB-S	–	Digital Video Broadcasting - Satellite
DVB-T	–	Digital Video Broadcasting - Terrestrial
EBU	–	European Broadcasting Union
EIT	–	Event Information Table
ELG	–	European Launching Group
EPG	–	Electronic Program Guide
ETSI	–	European Telecommunications Standards Institute
HDTV	–	High-Definition Television
HTML	–	HyperText Markup Language
LDTV	–	Low-Definition Television
LNB	–	Low-Noise Block converter
MHP	–	Multimedia Home Platform
NIT	–	Network Information Table
PAT	–	Program Association Table
PES	–	Packetized Elementary Stream
PHP	–	Hypertext Preprocessor
PID	–	Packet Identifier
PMT	–	Program Map Table
PS	–	Program Stream
PSI	–	Program-Specific Information
RST	–	Running Status Table
SDT	–	Service Description Table
SI	–	Service Information
SIT	–	Selection Information Table
ST	–	Stuffing Table

TDT	–	Time Data Table
TOT	–	Time Offset Table
TS	–	Transport Stream
WPS	–	Wireless Provisioning Services
WSS	–	Widescreen signaling
XML	–	Extensible Markup Language
SDTV	–	Standard-Definition TeleVision

Obsah

Seznam tabulek

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Analogové technologie jsou postupně vytlačovány digitálními. Toto se nevyhnulo ani televiznímu vysílání. Digitalizace přinesla nejen zkvalitnění služeb, ale i množství nových. Jedním z prvních typů digitálního vysílání bylo satelitní digitální vysílání. Satelitní vysílání přineslo hlavně zvýšení počtu vysílaných programů, dále také datové služby a v neposlední řadě i televizi s vysokým rozlišením (HDTV). Satelitní digitální vysílání není jediným typem digitálního vysílání, později vzniklo i kabelové a dále také pozemní a nakonec digitální vysílání pro mobilní zařízení.

Jak již z názvu této práce vyplývá, cílem je vytvořit aplikaci, která bude získávat EPG data z digitálního satelitního vysílání. EPG data jsou právě jednou z datových služeb umožněných digitalizací. V rámci této práce bude tedy vypracována aplikace pro sběr těchto dat z vysílání, jejíž funkcionalita bude detailně popsaná. Tato aplikace bude naimplementována jako služba, která poběží na pozadí systému. Dále bude v rámci práce vytvořená jednoduchá aplikace pro reprezentaci těchto dat. Funkcionalita této aplikace bude rovněž detailně popsána v rámci této práce. Zdrojové kódy aplikací budou volně přístupné na vybraném serveru[1].

Práce je rozvržena do pěti kapitol. První kapitolou je úvod. Kapitola druhá nese název *Digital Video Broadcasting*. Tato kapitola je složena z pěti dílčích částí. V první části je základní popis technologie DVB. Druhá část je věnována multiplexování, které umožňuje skládání více kanálů do jednoho. Ve třetí části jsou popsány servisní tabulky, do kterých jsou strukturovány servisní data. Čtvrtá část se věnuje deskriptorům, které jsou přenášeny v rámci servisních tabulek a obsahují z pravidla rozsáhlejší data. Poslední část druhé kapitoly se zabývá EPG daty. Třetí kapitola nese název *Možnosti řešení*. Kapitola je složena ze dvou dílčích částí. V první části jsou porovnány jednotlivé programovací jazyky vhodné pro řešení této práce, zejména pak C++ a Java. V druhé části jsou popsány použité knihovny, přičemž nejdůležitější je právě Linux DVB API, které obstarává řízení DVB karty.

Čtvrtá kapitola práce je věnována samotnému řešení. Jedná se o nejrozsáhlejší kapitolu práce. Tato kapitola je složena ze čtyř částí. První část je věnována vývoji aplikace pro získávání EPG dat z DVB-S/S2 systému. Je zde podrobně rozebrána funkcionalita jednotlivých částí aplikace. Další podkapitola se věnuje vývoji webového rozhraní pro prezentaci získaných EPG dat. Je zde proveden návrh a také podrobně popsaná funkcionalita aplikace. Následující část se zabývá testováním, je zde popsán postup a výsledky testování aplikací a také řešení nalezených nedostatků. Předposlední podkapitola se věnuje možnostmi dalšího rozšíření a vylepšení aplikace pro získávání EPG informaci z DVB-S/S2 systému. Poslední podkapitola popisuje použití jednotlivých aplikací.

[1] Zdrojové kódy je možné stáhnout z následujícího odkazu `https://github.com/Lukas951/Diploma`.

2 Digital Video Broadcasting

Z kraje 90. let vznikla skupina European Launching Group (ELG), která se zabývala problematikou zavedení digitální televize v celé Evropě. Tato skupina byla později přejmenována na Digital Video Broadcasting Project (DVB). DVB Project je konsorcium více jak 200 firem (původně pouze evropského původu, nyní ale po celém světě), kde od roku 1994 patří i Česká Televize. Specifikace DVB vydává Evropský telekomunikační institut ETSI ve spolupráci s Evropskou unií pro televizní a rozhlasové vysílání EBU.

DVB dnes nabízí několik úrovní kvality jak obrazu tak i zvuku, od televize s nízkou rozlišovací schopností LDTV přes standardní televizi SDTV až po televizi s vysokou rozlišovací schopností HDTV, v případě kvality zvuku pak od monofonního přes stereofonní až po zvuk 5.1 respektive Dolby Digital.

Bylo specifikováno několik standardů:

DVB-S Standard digitálního televizního vysílání přes satelitní systémy.

DVB-T Standard digitálního televizního vysílání přes pozemní vysílače.

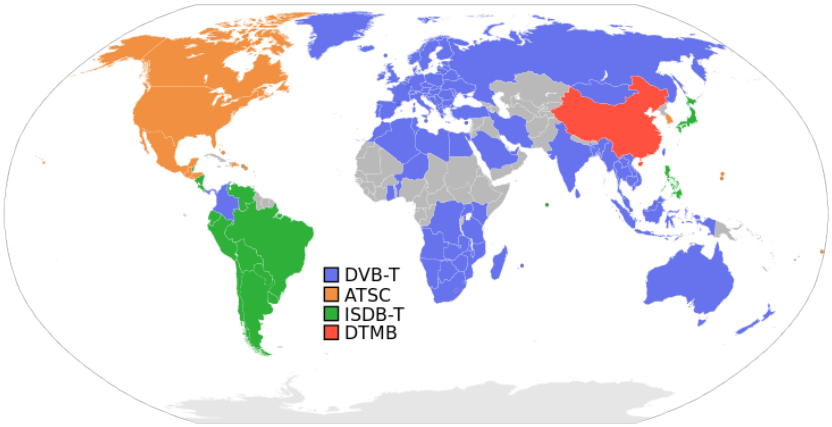
DVB-C Standard digitálního televizního vysílání v sítích kabelových televizí.

K těmto standardům později vznikly novější verze DVB-T2, DVB-S2 a také standard DVB-H, který specifikuje digitální vysílání pro mobilní zařízení. Použití různých standardů pro digitální vysílání ve světě, ilustruje obrázek 1. Z obrázku je patrné, že standard DVB-T se dá považovat za nejrozšířenější.

V roce 2000 byla DVB Projektem schválená tzv. konvergence ve vysílání mobility a multimédií. Od tohoto roku se specifikace DVB rozšířily na mobilní příjem v automobilech, mobilní příjem v přijímačích typu mobilní telefon (DVB-H), set-top boxy založené na řešení pomocí softwaru, interaktivní televizi, širokopásmový internet, otevřené normy platformy multimediálních domácích zařízení MHP. [4]

2.1 Popis technologie

Jedním z hlavních důvodů přechodu na digitální vysílání byla skutečnost lepšího využití frekvenčního spektra. Programy jsou převáděny v reálném čase do datového toku a následně společně komprimovány do formátu MPEG-2 (výjimečně do formátu MPEG-4, který je dokonalejší). Prakticky to znamená, že digitální vysílání umožňuje na jednom kanále, ve srovnání s analogovým vysíláním, přenést hned několik televizních stanic současně (u analogového vysílání sloužil jeden vysílací kanál pouze pro jednu televizní stanici). Skupina těchto kanálů se nazývá multiplex, který může krom skupiny televizních stanic obsahovat i několik doplňkových služeb (EPG, superteletext, atp). Pro tuto práci je nejvýznamnější právě EPG (Electronic Program Guide, Elektronický programový průvodce), kterému se věnuje kapitola 2.5.



Obrázek 1: DVB ve světě [9]

2.2 Multiplexování

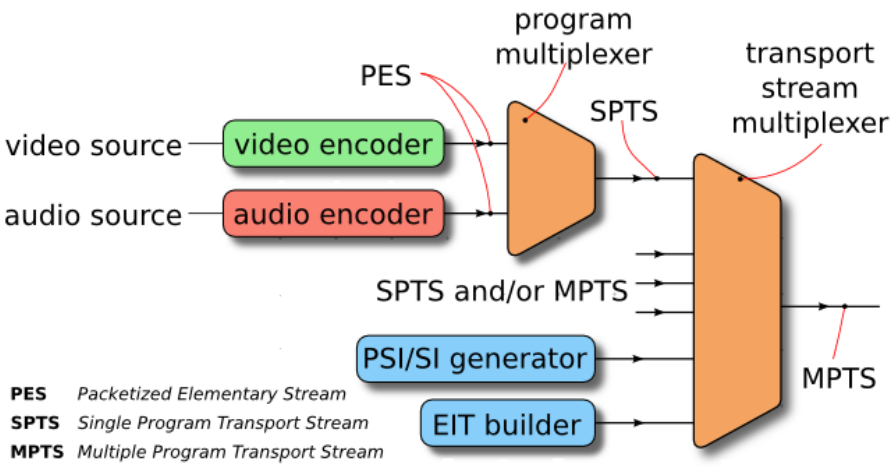
Jedná se o princip, který umožňuje vysílat více složek na jednom kanále. Aby bylo možné vysílat více složek na jednom kanále, je nutné tyto složky sloučit. Toto sloučení se nazývá multiplexování a provádí ho zařízení zvané multiplexer.

Jednotlivé zdrojové datové toky (PES - Packetized Elementary Stream) jsou nejprve vedeny do primárního neboli programového multiplexeru. Tyto datové toky obvykle představují zakódované obrazové a zvukové signály generované audio/video kodérem. Kromě těchto datových toků jsou do multiplexeru vedeny i doplňkové datové služby (teletext, podtitulky, WSS, WPS atd). Výstupem z programového multiplexeru je tzv. programový tok (PS - Program Stream). V praxi je obvykle videokodér, audiokodér a primární multiplexer tvořen jediným zařízením, zvaným DVB kodér.

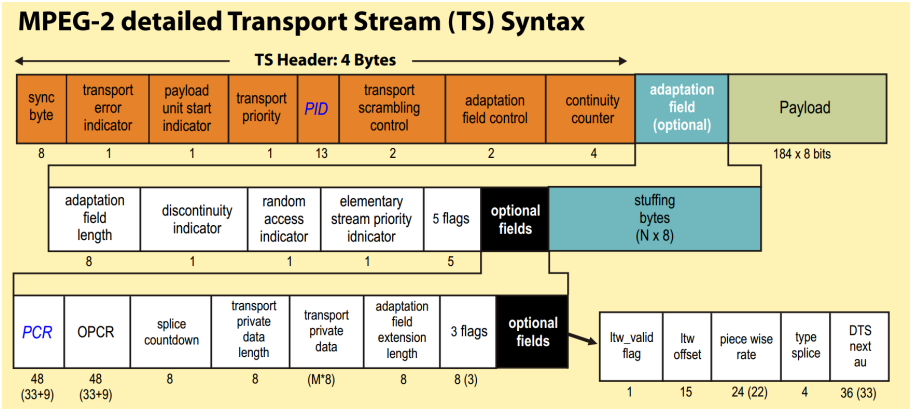
Výstupní programový datový tok primárního multiplexeru je dále veden do sekundárního multiplexeru neboli transportního multiplexeru. Zde probíhá sloučení jednotlivých programových toků a doplňkových dat, jako jsou například EPG, datový karusel, interaktivní aplikace apod., do jednoho transportního toku (TS - Transport Stream). Kromě toho transportní multiplexer vytváří servisní data, která identifikují jednotlivé složky výsledného transportního toku. Tato servisní data jsou uspořádaná do tabulek, které jsou přehledně zobrazeny v tabulce 1. Princip multiplexování ilustruje obrázek 2. [7]

2.2.1 Transport Stream Paket

Jednotlivé pakety TS mají délku 188 bytů. Každý TS začíná hlavičkou o délce 4 byty. První z nich tvoří synchronizační byte. Další důležitá informace uložená v hlavičce paketu je identifikační číslo paketu tzv. PID (Packet Identification). Zbylých 184 bytů paketu jsou samotná data. Kompletní strukturu TS paketu ilustruje obrázek 3.



Obrázek 2: Princip dvb multiplexování [1]



Obrázek 3: Struktura TS packetu [12]

2.3 Servisní informační tabulky

Servisní informační tabulky specifikuje norma EN 300 468 [3]. Tabulky jsou vysílané v sekcích. Jednotlivé tabulky a jejich PID hodnoty jsou shrnuty v tabulce 1. Funkce jednotlivých tabulek jsou shrnuty v následujícím výpisu:

Program Association Table (PAT) Pro každou službu v multiplexu, PAT označuje umístění příslušné PMT a NIT.

Conditional Access Table (CAT) CAT poskytuje informace o systémech CA používaných v multiplexu.

Program Map Table (PMT) Podrobné informace o jednom programu a jeho složkách.

Network Information Table (NIT) Nese informace o síti samotné.

Bouquet Association Table (BAT) Poskytuje informace týkající se pugetů.

Service Description Table (SDT) Nese informace o službách v určitém TS.

Event Information Table (EIT) Nese informace o vysílaných událostech (pořadech).

Running Status Table (RST) Umožňuje přesné a rychlé aktualizace načasování jednotlivých událostí.

Time and Date Table (TDT) Nese pouze UTC datum a čas.

Time Offset Table (TOT) Nese UTC datum, čas a místní časový posun.

Stuffing Table (ST) Účelem této tabulky je rušení existujících sekcí na hranicích systému.

Selection Information Table (SIT) Používá se pouze v „částečných“ (tj. zaznamenaných) tocích. Nese přehled SI informací požadovaných pro popis toků v částečném bitovém toku.

Discontinuity Information Table (DIT) Používá se pouze v „částečných“ (tj. zaznamenaných) tocích. Je vkládána tam, kde SI informace v částečných bitových tocích může být přerušovaná.

Každá tabulka má identifikátor table_id, ale jinak se tabulky svou strukturou liší. Detailní popis struktur jednotlivých tabulek poskytuje norma EN 300 468. Většina důležitých informací je uchována v descriptorech jednotlivých tabulek. Tyto descriptory budou popsány dále. Tabulky využívané v této práci budou popsány podrobněji. [3]

Table	PID value hex	PID value dec
PAT	0x0000	0
CAT	0x0001	1
TSDT	0x0002	2
reserved	0x0003 to 0x000F	3 to 15
NIT, ST	0x0010	16
SDT, BAT, ST	0x0011	17
EIT, ST	0x0012	18
RST, ST	0x0013	19
TDT, TOT, ST	0x0014	20
network synchronization	0x0015	21
reserved for future use	0x0016 to 0x001D	22 to 29
DIT	0x001E	30
SIT	0x001F	31

Tabulka 1: Tabulka pro jednotlivé PID[3]

2.3.1 Service Description Table

Každá sekce SDT popisuje služby které obsahuje konkrétní TS (službou se zde rozumí například jednotlivé televizní programy jako je ČT 1). SDT se člení do service_description_sections podle syntaxe zobrazené na obrázku 4. Všechny části, které jsou součástí SDT, se vysílají v TS paketech s hodnotou PID 0x0011. Každá sekce, která popisuje aktuální TS, má hodnotu table_id 0x42, a každá sekce SDT, která odkazuje na jiný než aktuální TS, má hodnotu table_id 0x46. [3]

2.3.1.1 Popis jednotlivých atributů SDT [3]

table_id: viz tabulka 2.

section_syntax_indicator: jednobitová hodnota, která musí být nastavena na "1".

section_length: Délka sekce v bytech.

transport_stream_id: Hodnota pro identifikaci TS.

version_number: Hodnota verze sub_table.

current_next_indicator: Jednobitová hodnota označující platnost sub_table.

section_number: Číslo aktuální sekce.

last_section_number: Číslo poslední sekce v sub_table.

original_network_id: Označení určující network_id původního doručujícího systému.

service_id: Identifikační číslo služby.

Syntax	No. of bits	Identifier
service_description_section(){		
table_id	8	uimsbf
section_syntax_indicator	1	bslbf
reserved_future_use	1	bslbf
reserved	2	bslbf
section_length	12	uimsbf
transport_stream_id	16	uimsbf
reserved	2	bslbf
version_number	5	uimsbf
current_next_indicator	1	bslbf
section_number	8	uimsbf
last_section_number	8	uimsbf
original_network_id	16	uimsbf
reserved_future_use	8	bslbf
for (i=0;i<N;i++){		
service_id	16	uimsbf
reserved_future_use	6	bslbf
EIT_schedule_flag	1	bslbf
EIT_present_following_flag	1	bslbf
running_status	3	uimsbf
free_CA_mode	1	bslbf
descriptors_loop_length	12	uimsbf
for (j=0;j<N;j++){		
descriptor()		
}		
}		
CRC_32	32	rpchof
}		

Obrázek 4: Service description sections[3]

EIT_schedule_flag: Jednobitová hodnota určující, zda jsou EIT informace aktuální služby dostupné v aktuálním TS.

EIT_present_following_flag: Aktuální služby dostupné v aktuálním TS.

running_status: Hodnota určující status služby.

free_CA_mode: Jednobitová hodnota určující zda je služba zakódovaná či nikoli.

descriptors_loop_length: Celková délka deskriptoru v bytech.

2.3.2 Event Information Table

EIT poskytuje informace v chronologickém pořadí vzhledem k pořadům obsaženým v rámci každé služby. EIT je možné klasifikovat do čtyř skupin podle hodnoty table_id:

1. aktuální TS, present/following informace o pořadu = table_id = "0x4E",
2. ostatní TS, present/following informace o pořadu = table_id = "0x4F",
3. aktuální TS, event schedule informace o pořadech = table_id = "0x50"to "0x5F",
4. ostatní TS, event schedule informace o pořadech = table_id = "0x60"to "0x6F".

Value	Description
0x00	program_association_section
0x01	conditional_access_section
0x02	program_map_section
0x03	transport_stream_description_section
0x04 to 0x3F	reserved
0x40	network_information_section - actual_network
0x41	network_information_section - other_network
0x42	service_description_section - actual_transport_stream
0x43 to 0x45	reserved for future use
0x46	service_description_section - other_transport_stream
0x47 to 0x49	reserved for future use
0x4A	bouquet_association_section
0x4B to 0x4D	reserved for future use
0x4E	event_information_section - actual_transport_stream, present/following
0x4F	event_information_section - other_transport_stream, present/following
0x50 to 0x5F	event_information_section - actual_transport_stream, schedule
0x60 to 0x6F	event_information_section - other_transport_stream, schedule
0x70	time_date_section
0x71	running_status_section
0x72	stuffing_section
0x73	time_offset_section
0x74 to 0x7D	reserved for future use
0x7E	discontinuity_information_section
0x7F	selection_information_section
0x80 to 0xFE	user defined
0xFF	reserved

Tabulka 2: Hodnoty jednotlivých table_id[3]

Syntax	No. of Bits	Identifier
event_information_section(){		
table_id	8	uimsbf
section_syntax_indicator	1	bslbf
reserved_future_use	1	bslbf
reserved	2	bslbf
section_length	12	uimsbf
service_id	16	uimsbf
reserved	2	bslbf
version_number	5	uimsbf
current_next_indicator	1	bslbf
section_number	8	uimsbf
last_section_number	8	uimsbf
transport_stream_id	16	uimsbf
original_network_id	16	uimsbf
segment_last_section_number	8	uimsbf
last_table_id	8	uimsbf
for(i=0;i<N;i++){		
event_id	16	uimsbf
start_time	40	bslbf
duration	24	uimsbf
running_status	3	uimsbf
free_CA_mode	1	bslbf
descriptors_loop_length	12	uimsbf
for(i=0;i<N;i++){		
descriptor()		
}		
}		
CRC_32	32	rpchof
}		

Obrázek 5: Event information section[3]

Tabulka informací present/following obsahuje pouze informace týkající se pouze současného pořadu a chronologicky následujícímu patřícímu k dané službě buď na aktuálním TS nebo na ostatních TS. Tabulka event schedule buď pro aktuální nebo ostatní TS, obsahuje list událostí ve formě harmonogramu, které se konají v určité době za následujícím pořadem.

EIT je rozdělena do event_information_sections podle syntaxe zobrazené na obrázku 5. Všechny části, které jsou součástí EIT se vysílají v TS paketech s hodnotou PID 0x0012. [3]

2.3.2.1 Popis jednotlivých atributů EIT [3]

table_id: viz tabulka 2.

section_syntax_indicator: Jednobitová hodnota, která musí být nastavena na "1".

section_length: Délka sekce v bytech.

transport_stream_id: Hodnota pro identifikaci TS.

version_number: Hodnota verze sub_table.

current_next_indicator: Jednobitová hodnota označující platnost sub_table.

section_number: Číslo aktuální sekce.

Deskriptor	Tag	SI/PSI deskriptory					
		NIT	**BAT**	**SDT**	**EIT**	**TOT**	**PMT**
network_name_descriptor	0x40	*	-	-	-	-	-
service_list_descriptor	0x41	*	*	-	-	-	-
service_descriptor	0x48	-	-	*	-	-	-
short_event_descriptor	0x4D	-	-	-	*	-	-
extended_event_descriptor	0x4E	-	-	-	*	-	-
content_descriptor	0x54	-	-	-	*	-	-
teletext_descriptor	0x56	-	-	-	-	-	*

Tabulka 3: Tabulka vybraných deskriptorů[3]

last_section_number: Číslo poslední sekce v sub_table.

original_network_id: Označení určující network_id původního doručujícího systému.

segment_last_section_number: Číslo posledního segmentu.

last_table_id: Hodnota posledního použitého table_id.

event_id Hodnota identifikující událost.

start_time Hodnota určující čas začátku pořadu v UTC.

duration Hodnota určující délku trvání pořadu.

format

running_status: Hodnota určující status služby.

free_CA_mode: Jednobitová hodnota určující zda je služba zakódovaná či nikoli.

descriptors_loop_length: Celková délka deskriptoru v bytech.

2.4 Deskriptory

Jak již bylo zmíněno, jsou servisní informace vysílané v rámci TS strukturovány do tabulek. Popis těchto tabulek proběhl v části 2.3. Tyto tabulky také obsahuji tzv. deskriptory, které se používají pro uchování obsáhlejších informací. Tabulka 3 obsahuje pouze výběr některých deskriptorů, z nichž většina byla použita v této práci. Kompletní seznam deskriptorů je definovaný v normě EN 300 468 [3].

Jednotlivé deskriptory se skládají z hlavičky a těla. V hlavičce se nachází tag deskriptoru a jeho délka. Tělo deskriptoru obsahuje užitečné informace a je pro každý typ deskriptoru různé. Právě tento typ určuje tag deskriptoru. Deskriptory využívané v této práci budou popsány podrobněji v následujících odstavcích.

Syntax	No. of bits	Identifier
service_descriptor(){		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
service_type	8	uimsbf
service_provider_name_length	8	uimsbf
for (i=0;i<N;i++){		
char	8	uimsbf
}		
service_name_length	8	uimsbf
for (i=0;i<N;i++){		
char	8	uimsbf
}		
}		

Obrázek 6: Service descriptor[3]

Syntax	No. of bits	Identifier
short_event_descriptor(){		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
ISO_639_language_code	24	bslbf
event_name_length	8	uimsbf
for (i=0;i<event_name_length;i++){		
event_name_char	8	uimsbf
}		
text_length	8	uimsbf
for (i=0;i<text_length;i++){		
text_char	8	uimsbf
}		
}		

Obrázek 7: Short event descriptor[3]

2.4.1 Service descriptor

Service descriptor (deskriptor služby) obsahuje název služby (název programu - ČT 1, NOVA atp.) a název poskytovatele (např. České radiokomunikace). Další informaci přenášenou v deskriptoru je typ služby. Syntaxe tohoto deskriptoru je zobrazena na obrázku 6. [3]

2.4.2 Short event decriptor

Short event deskriptor (krátký deskriptor události) obsahuje název vysílaného pořadu a jeho krátký popis. Tyto informace jsou součástí EPG. Syntaxe tohoto deskriptoru je zobrazena v obrázku 7. [3]

2.4.3 Extended event decriptor

Extended event descriptor (rozšířený deskriptor události) obsahuje detailní popis pořadu, který může být použit navíc k short event deskriptoru. Těchto deskriptorů může být více pro jeden pořad v případě popisu delšího než 256 bytu. Dále obsahuje číslo aktuálního deskriptoru a číslo posledního deskriptoru, které mají využití právě v případě delšího popisu než 256 bytu. Další obsaženou položkou v deskriptoru je kód země re-

Syntax	No. of bits	Identifier
extended_event_descriptor(){		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
descriptor_number	4	uimsbf
last_descriptor_number	4	uimsbf
ISO_639_language_code	24	bslbf
length_of_items	8	uimsbf
for (i=0;i<N;i++){		
item_description_length	8	uimsbf
for (j=0;j<N;j++){		
item_description_char	8	uimsbf
}		
item_length	8	uimsbf
for (j=0;j<N;j++){		
item_char	8	uimsbf
}		
}		
text_length	8	uimsbf
for (i=0;i<N;i++){		
text_char	8	uimsbf
}		
}		

Obrázek 8: Extended event descriptor[3]

Syntax	No. of bits	Identifier
content_descriptor(){		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
for (i=0;i<N;i++) {		
content_nibble_level_1	4	uimsbf
content_nibble_level_2	4	uimsbf
user_nibble	4	uimsbf
user_nibble	4	uimsbf
}		
}		

Obrázek 9: Content descriptor[3]

spektive jazyku. Tyto informace jsou součásti EPG. Syntaxe tohoto deskriptoru je zobrazena v obrázku 8. [3]

2.4.4 Content descriptor

Záměrem content deskriptoru je poskytovat informace pro klasifikaci jednotlivých pořadů. Tyto informace jsou zakódovány pomocí dvou 4 bytových čísel. První číslo specifikuje, zdali se jedná o zprávy, film, seriál, sport atd. Druhé číslo upřesňuje daný pořad (například u filmu určuje, zdali se jedná o thriller, komedii atp.). Jednotlivé kódování klasifikace pořadů obsahuje tabulka 5 v příloze A. Syntaxe tohoto deskriptoru je zobrazena v obrázku 9. [3]

2.5 Electronic program guide

Electronic program guide neboli elektronický programový průvodce, je standardní služba pro digitální vysílání umožnující zobrazit informace o vysílaném obsahu. Jedná se v pod-

Akcent	Kód	Druhý znak	Výsledek
čárka	C2	AEIOUYaeiouy	ÁÉÍÓÚÝáéíóúý
kroužek	CA	AUau	ÅŮåů
háček	CF	CDENRSTZcdenrstz	ČĎĚŇŘŠŤŽčďěňřšťž

Tabulka 4: Tabulka pro kódování diakritiky podle standartu ISO 6937

statě o elektronický (nejen) televizní program. Základní provedení EPG bez požadavků na speciální middleware je definován normou EN 300 468. EPG data definovaná standardem jsou přenášena pomocí tabulek EIT. Součástí EPG dat jsou i informace o programech, které jsou uloženy v tabulkách SDT. Detailní popis těchto tabulek proběhl v části 2.3.2.

Pro ukládání EPG dat se používá XMLTV formát. V tomto formátu se využívá pouze dvou záznamů a to *channels* a *programme*. Jak už z názvu vyplývá, je záznam *channels* používaný pro ukládaní dat o programu a záznam *programme* pro ukládání dat o pořadech. Většina informací o pořadech/programech je nepovinná a nemusí být dostupná u všech zdrojů. Základní strukturu XMLTV formátu zobrazuje výpis 1. [8]

```
<tv>
    <channel id="257">
        display-name>CT 1</display-name>
    </channel>
    .
    .
    .
    <programme channel="16647" start="20150402043000 +0200" stop="20150402045000 +0200"
        >
        < title  lang="cs">Svet ve 20 minutach</title>
        <sub-title lang="cs">Zajimave clanky ze svetovych medii.</sub-title>
    </programme>
    .
    .
    .
</tv>
```

Výpis 1: Struktura XMLTV

Pro kódování českých znaků je použit standard ISO 6937. Tento standard je zkonstruován jako vícebytové rozšíření ANSCI. Některé kódy byly použity pro diakritické znaky (výběr základních znaků pro českou diakritiku je zobrazen v tabulce 4).

3 Možnosti řešení

Tato část práce se zabývá možnostmi řešení, jako je porovnání programovacích jazyků, jejich výhody či nevýhody. Jsou sou zde také zahrnuty použité knihovny a jejich stručný popis.

3.1 Porovnání jazyků

Při výběru programovacího jazyka bylo nutné vzít v úvahu požadavky kladené v rámci zadání. Jedním z hlavních kritérií bylo omezení na operační systém Linux. Tímto byla množina programovacích jazyků zúžena na následující: C, C++, JAVA.

Jazyk JAVA je plně objektově orientovaný a disponuje oproti jazykům C/C++ automatickým správcem paměti tzn. garbage collector a zjednodušenou syntaxí. Pro svou funkci ale potřebuje interpret tzn. virtuální stroj. Nutnost mít nainstalovaný virtuální stroj v systému se stala klíčovou při výběru implementačního jazyka, jelikož běh programu ve virtuálním stroji omezuje přístup k hardwaru. Dalším důležitým kritériem při rozhodování byl jazyk knihoven pro řízení DVB zařízení v operačním systému Linux.

Po důkladném zvážení byl jazykem pro implementaci aplikace zvolen jazyk C++, který oproti jazyku C přináší několik inovací, jako je například možnost objektového programování.

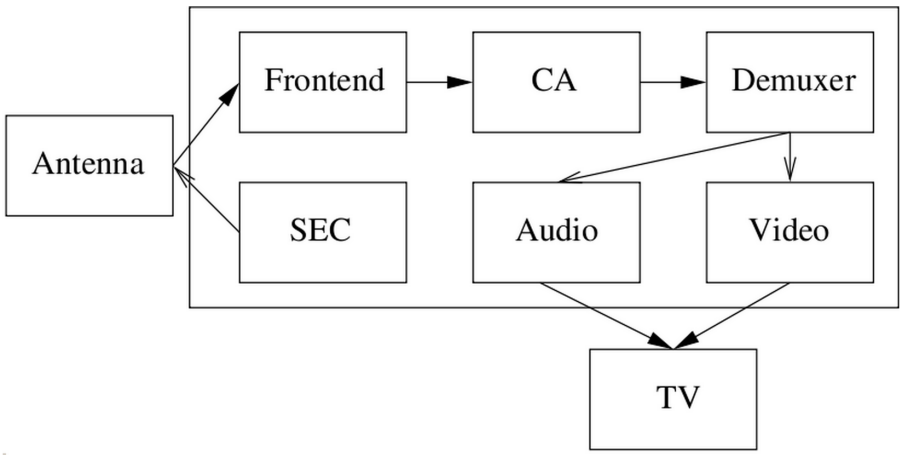
3.2 Knihovny

Tato část se zabývá použitými knihovnami v diplomové práci. Základní sadou pro řízení a práci s DVB kartou pod operačním systémem Linux je Linux DVB API. Dalším důležitým prvkem je balík knihoven a utilit dvb-apps.

3.2.1 Linux DVB API [5]

Linux DVB API se používá pro řízení DVB karet pod operačním systémem Linux. DVB karty se obvykle skládají z těchto hardwarových komponent:

- **Frontend skládající se z tuneru a demodulátoru**
  Frontend zařízení přijímá signál ze satelitu či antény nebo přímo z kabelu. Zařízení konvertuje a demoduluje signál do MPEG datového proudu (TS). V případě satelitu, frontend zahrnuje zařízení pro ovládání satelitní techniky (SEC), které umožňuje kontrolu LNB polarizace, vice-zdrojových přepínačů nebo rotaci paraboly.
- **Conditional Access (CA) hardware jako jsou CI adaptéry a sloty pro inteligentní karty**
  Kompletní TS prochází přes CA hardware. Programy, na které má uživatel přístup (řízené inteligentními kartami), jsou dekódovány v reálném čase znovu vloženy do TS.



Obrázek 10: Komponenty DVB karet[5]

- **Demultiplexer, který filtruje příchozí datový proud**
  Demultiplexer rozdělí TS do jeho komponent, jako jsou datové toky zvuku a videa. Kromě obvykle několika takových zvukových a obrazových toků rovněž obsahuje datové toky s informací o nabízených programech v tomto nebo jiném proudu stejného poskytovatele.

- **MPEG2 zvukový a video dekodér**
  Demultiplexer rozešle audio a video datové toky do příslušných MPEG2 dekodérů. Po dekódování přecházejí jako nekomprimované audio a video na obrazovku počítače nebo (prostřednictvím PAL / NTSC enkodéru) na obrazovku televizoru.

Obrázek 10 zobrazuje hrubé schéma datových a ovládacích toků mezi komponentami DVB karet.

Linux DVB API umožňuje tyto hardwarové komponenty ovládat prostřednictvím šesti unixových zařízení pro video, zvuk, frontend, demux, CA a IP-over-DVB sítě. Zařízení pro video a zvuk ovládá MPEG-2 dekódovací hardware, frontend zařízení ovládá tuner a DVB demodulátor. Zařízení demux umožňuje ovládat PES a filtry sekcí pomocí hardware. Pokud hardware filtrování nepodporuje, může byt implementováno programově. Zařízení CA řídí hardware podmíněného přístupu.

3.2.2 Dvb-apps [6]

Jedná se o balík knihoven a sadu utilit užitečných jak pro vývojáře, tak pro konečného uživatele. Pro tuto práci jsou důležité zejména tyto knihovny:

- **Knihovna libucsi**
  Tato knihovna je určená pro parsování SI tabulek. Proto se nabízí její použití při dekódování tabulek tvořících EPG data.

- **Knihovna libdvbcfg**
  Tato knihovna umožňuje vytvářet a parsovat konfigurační soubory digitálních kanálů. Pro tuto práci je zajímavá funkce dvbcfg_zapchannel_parse, která provádí parsování konfiguračního souboru.

Dvb-apps nabízí i několik zajímavých utilit, zejména pak scan, dvbscan a `t/c/s zap`. První dvě jmenované utility se používají pro generování konfiguračního souboru pro dostupné kanály. Poslední ze jmenovaných utilit se používá pro ladění jednotlivých kanálů.

4 Řešení

Tato kapitola se bude věnovat samotnému implementování aplikací. Jedná se o aplikaci pro získání EPG dat, které má běžet jako trvalá služba pod systémem Linux a dále o webové aplikaci pro zobrazení získaných EPG dat. Bude zde popsán jak princip a funkce jednotlivých aplikací, tak také okomentovaný průběh a výsledky testování. Dále zde bude zmíněno několik možností vylepšení respektive rozšíření aplikace.

Jak již bylo zmíněno v kapitole 3.1, hlavní aplikace jakožto aplikace pro získání EPG dat bude implementována pomocí jazyka C++. Pro aplikaci zobrazující získané EPG data bude využito jazyků HTML, JavaScript a PHP.

4.1 Aplikace pro získávání EPG informací z DVB-S systému

V rámci této podkapitoly proběhne popis vývoje aplikace pro získávání EPG informací z DVB-S/S2 systému. Bude zde proveden návrh logické funkcionality aplikace a dále budou detailně popsány jednotlivé funkční celky. Tyto celky jsou: parsování konfiguračního souboru, ladění, nastavení DiSEqC, získání dat, zpracování dat a export dat do XML.

4.1.1 Linux daemon

Linux démon je označení pro službu v operačním systému Linux. Jelikož jedním z požadavků zadání bylo naimplementovat aplikaci, aby mohla běžet jako trvalá služba, proto bylo nutné zohlednit tento požadavek jak v návrhu, tak i v implementaci samotné.

Démon je tedy typ programu, který běží nenápadně v pozadí systému. Démon se spouští buď samotným systémem, spouštěcím skriptem nebo uživatelem přes terminál. Příkladem takového programu může být například webový server. [10]

Základní struktura démona je:

- **Rozvětvení rodičovského procesu**
  K tomu je určena funkce *fork()*, která vrací celočíselnou hodnotu reprezentující získaný PID.
- **Změna masky práv nově vytvořených souborů (umask)**
  Aby bylo možné zapisovat a číst ze souboru vytvořených démonem, je nutné změnit režim masky souboru. K tomuto účelu je určená funkce *umask()*. Nastavením *umask(0)* získáme plný přístup k souborům vytvořeným démonem.
- **Otevření log souborů pro zápis**
  Tato část je volitelná, jsou zde otevřeny jednotlivé log soubory jako je syslog pro zápis.
- **Vytvoření unikátního Session ID (SID)**
  Proces musí získat od rodičovského procesu unikátní SID. K tomuto je určená funkce *setsid()*, která vrací celočíselnou hodnotu reprezentující získaný SID.

- **Změna pracovního adresáře**
  Pracovní adresář by měl být změněn na místo, které zde bude vždy přítomné. Jelikož ne všechny distribuce Linuxu úplně dodržují standardní hierarchii souborového systému, jediným takovým adresářem je *root "/"*. Funkce provádějící tuto změnu je*chdir("/")*.

- **Zavření standardních file deskriptorů**
  Jedním z posledních kroků v nastavení démona je zavření standardních deskriptorů souborů (STDIN, STDOUT, STDERR). Vzhledem k tomu, že démon nemůže používat terminál, jsou tyto deskriptory souborů nadbytečné. Využívá se funkce *close()*.

- **Samotný kód démona**
  Většinou je tvořen nekonečnou smyčkou (hlavní smyčka programu). [11]

4.1.2 Návrh aplikace

Při návrhu aplikace bylo nutné vycházet z požadavků daných zadáním. Aplikace by měla získávat EPG data vysílané na různých transpondérech družice. Na tyto transpondéry je nutné DVB kartu naladit. Informace o jednotlivých transpondérech jsou uloženy v konfiguračním souboru *channels.conf*, který je možné vygenerovat prostřednictvím utility *scan* obsažené v balíku dvb-apps. Detailnější popis balíku dvb-apps byl proveden v sekci 3.2.2. Základní použití utility *scan* je následující:

Pro systém DVB-S/S2: *scan /usr/share/dvb/dvb-s/Astra-23.5E > channels.conf*

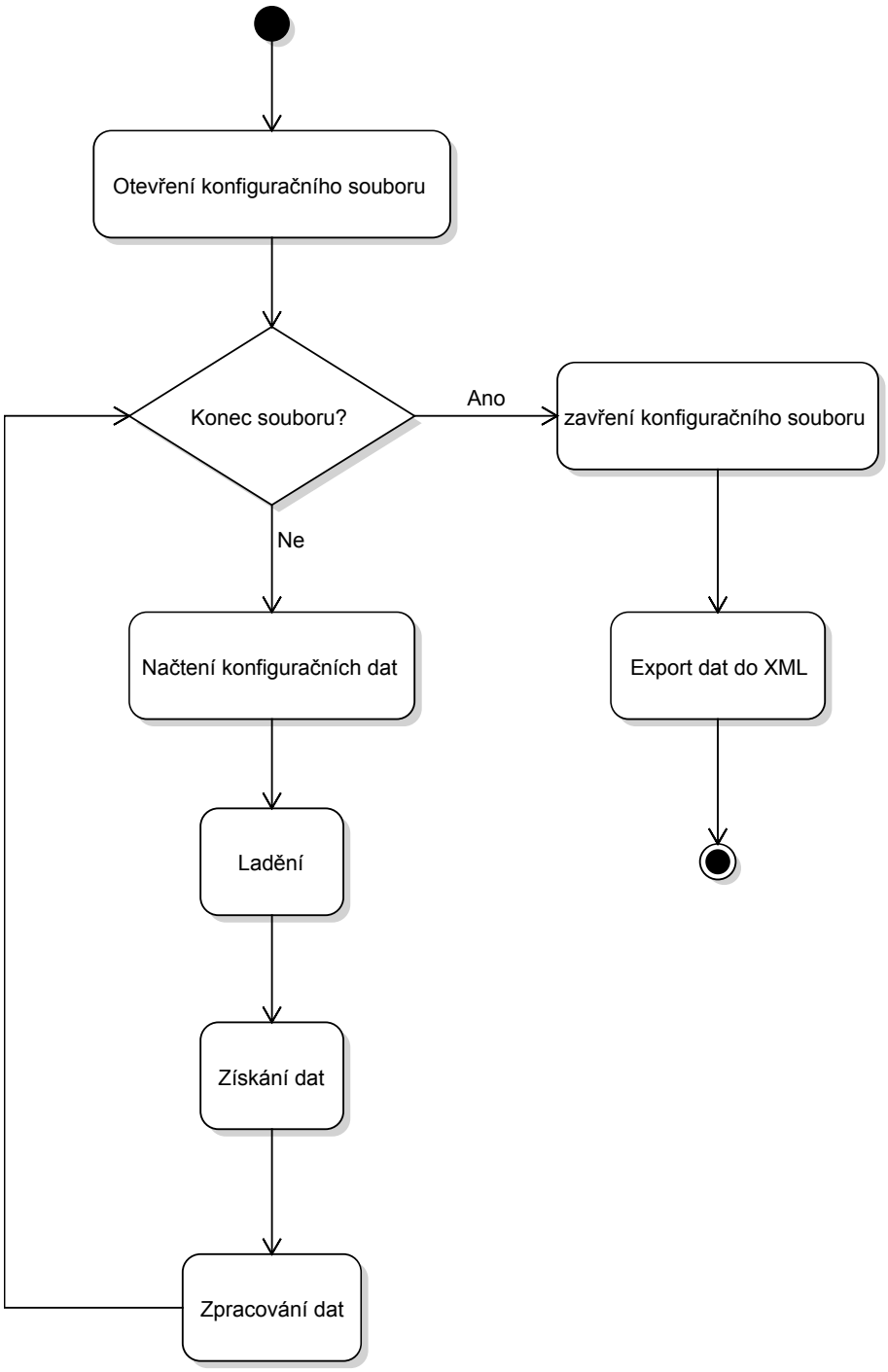
Pro systém DVB-T: *scan /usr/share/dvb/dvb-t/cz-All > channels.conf*

Takto vygenerovaný konfigurační soubor obsahuje informace důležité pro ladění pro dostupné programy, kde jeden řádek obsahuje údaje právě pro jeden program.

Aplikace by tedy měla rozparsovat řádek konfiguračního souboru, naladit podle získaných dat a následně zahájit sběr EPG informací z vysílaní. Získané informace je nutné zpracovat. Tímto zpracováním se zamýšlí rozkódování jednotlivých informací ze servisních tabulek, dále převod z formátu ISO 6937 do formátu UTF-8. Takto získaná a zpracována data je nutné uložit do vhodné struktury. Po získání a zpracování EPG dat by se mělo přejít k dalšímu řádku konfiguračního souboru a proces opakovat. Pro lepší ilustraci je tento proces zachycen v diagramu aktivit 11. Jelikož má být aplikace implementována jako trvalá služba, je v diagramu aktivit zachycena pouze jedna iterace hlavní smyčky programu.

Zde však nastává problém v určité „náhodnosti“ získaných EPG dat. DVB systém umožňuje z přijatých dat získat data s určitým PID, ale už není možné ovlivnit o jaké data se konkrétně jedná, tzn. některá data aplikace přijme víckrát a na jiná budu muset dlouho čekat.

Z tohoto důvodu je efektivnější nečekat na všechny EPG informace k jednomu programu, ale získávat EPG data jednoho kanálu pouze určitý čas a následně přeladit na



Obrázek 11: Diagram aktivit

další program, přičemž k předchozímu programu se aplikace vrátí v další iteraci hlavní smyčky. Určitým zrychlením by mohla být i paralelizace získávání dat.

4.1.3 Komunikace s DVB kartou

Pro komunikaci je využito Linux DVB API, které bylo detailněji popsané v kapitole 3.2.1. Základní funkce pro komunikaci jsou funkce *open, close, read, ioctl*. Tyto funkce je nutné detailně popsat, jelikož jsou často využívány v samotné aplikaci.

int open(const char *deviceName, int flags); Funkce přijímá dva parametry. Prvním parametrem je řetězec reprezentující cestu k zařízení (frontend, demux atd.). Druhý parametr určuje režim, v jakém se s zařízením pracuje (například readonly, read/write viz [5]). Funkce vrací parametr typu *int* reprezentující *File Descriptor*.

int close(int fd); Funkce přijímá parametr typu *int*, který reprezentuje *File Descriptor* získaný dřívějším zavolání funkce *Open*. Funkce ukončuje komunikaci se zařízením.

int read(int fd, void *buf, int count); Funkce má tři parametry. Prvním z nich je *File Descriptor* zařízení demux. Druhý parametr je pointr na buffer pro uložení získaných dat. Posledním parametrem je velikost bufferu. Funkce vrací hodnotu *int* reprezentující velikost získaných dat.

int ioctl(int fd, int request, ...); Tato funkce se používá pro nastavení a získání vlastností frontend zařízení. Pro toto využití přijímá funkce tři parametry. Prvním je *File Descriptor* určitého frontend zařízení, druhým parametrem je celočíselná hodnota reprezentující požadovanou operaci (např. *FE_SET_FRONTEND, FE_GET_FRONTEND*, viz [5]). Jako třetí parametr se předává určitá struktura zvolená podle typu operace. V případě zmíněných operací je to struktura *dvb_frontend_parameters*.

4.1.4 Parsování konfiguračního souboru

Jelikož jsou data o jednotlivých kanálech uložena v konfiguračním souboru s jistou strukturou, je nutné tento soubor rozparsovat. Pro parsování konfiguračního souboru je využitá knihovna *libdvbcfg*, která již byla zmíněná v sekci 3.2.2. Z této knihovny byla využitá funkce *dvbcfg_zapchannel_parse* viz výpis 2. Ve výpisu je ilustrováno právě volání této funkce. Tato funkce má tři parametry. První z nich reprezentuje konfigurační soubor, druhým je callback funkce, která se volá pro každý rozparsovaný řádek konfiguračního souboru. Posledním parametrem jsou privátní data, která jsou přeposlány až do callback funkce. V tomto případě je tento parametr nevyužit.

```
FILE *fchannels = fopen(channelsfile, "r");
if (fchannels == NULL)
{
    syslog(LOG_INFO, "Unable to open %s\n", channelsfile);
    exit(1);
}

dvbcfg_zapchannel_parse(fchannels, channels_cb, (void*) void);
```

Výpis 2: Volání funkce pro parsování konfiguračního souboru

Samotné parsování probíhá následovně. Nejprve je otevřen konfigurační soubor v režimu *read*. Následně je zavolána funkce *dvbcfg_zapchannel_parse* s příslušnými parametry. Funkce pak rozparsovává konfigurační soubor řádek po řádku. Rozparsovaná data jednoho řádku pak pomocí struktury *dvbcfg_zapchannel* předá callback funkci. Tato callback funkce je volaná pro každý rozparsovaný řádek zvlášť. V tomto případě slouží pro callback funkce *channels_cb*.

V rámci této funkce je převedena struktura *dvbcfg_zapchannel* z knihovny *libdvbcfg* na strukturu *dvb_frontend_parameters*, která je součásti Linux DVB API. Tento převod byl nutný z důvodu dalšího využití dat ve funkci pro ladění, která vyžaduje data ve struktuře *dvb_frontend_parameters*. Následně jsou získány file deskriptory pro frontend a demux zařízení pomocí funkce *open* s příslušnými parametry. Poté je provedeno zjištění typu zařízení pomocí funkce *ioctl*. Funkci je předán file deskriptor, dále parametr *FE_GET_INFO* a posledním parametrem je struktura pro návrat dat z funkce *dvb_frontend_info*. Takto je získána struktura obsahuje mimo jiné i požadované informace o zařízení. Na základě těchto informací a rozparsovaných dat z konfiguračního souboru je rozhodováno, jestli se jedná o zařízení DVB-S, popřípadě zařízení DVB-T. V případě zařízení DVB-T se předají parametry funkci pro ladění a je provedeno naladění a následné získání EPG dat. Pokud se jedná o zařízení DVB-S, je provedena korekce frekvence podle typu LNB. Poté se provádí nastavení zařízeni DiSEqC, naladění a získaní EPG dat.

4.1.5 Ladění

Pro samotné ladění byla naimplementována funkce *tune*. Funkce přijímá dva parametry. Prvním je file deskriptor *fe_fd* a druhým parametrem je struktura *dvb_frontend_parameters* obsahující data pro naladění. Naladění, respektive nastavení frontend zařízení, je provedeno pomocí funkce *ioctl*. Funkci je předán file deskriptor, dále parametr *FE_SET_FRONTEND* a dále struktura dat pro naladění. Struktura *dvb_frontend_parameters* obsahuje různá data podle typu zařízení (DVB-T, DVB-S, atd.), ale nastavení frontend zařízení se provádí stejně. Funkce je zobrazena ve výpisu 3.

```
int tune(int fe_fd, dvb_frontend_parameters p)
{

    if ( ioctl (fe_fd, FE_SET_FRONTEND, &p) < 0) {
        syslog(LOG_INFO, "FE_SET_FRONTEND failed");
        return -1;
    }

    return 1;

}
```

Výpis 3: Funkce tune

Jak již bylo zmíněno, nastavené parametry ve struktuře *dvb_frontend_parameters* se liší podle typu zařízení. Parametry sdílené pro všechny zařízení jsou *frequency* a *inversion*. Další parametry pak záleží na typu zařízení, v případě DVB-S to jsou: *symbol_rate, fec_inner*. V případě zařízení DVB-C se jedná o parametry: *symbol_rate, fec_inner, modulation*. Jestliže se jedná o zařízení DVB-T jsou parametry následující: *bandwidth, code_rate_HP, code_rate_LP, constellation, transmission_mode, guard_interval, hierarchy_information*. Přesnější popis těchto parametrů a možnosti nastavení popisuje dokument [5].

4.1.6 Nastavení DiSEqC

Pro nastavení DiSEqC jsou využity funkce *diseqc* a funkce *diseqc_send_msg*. Tyto funkce byly získané a následně upravené z balíků dvb-apps, konkrétně z utility szap, jejíž autorem je Johannes Stezenbach (2001). Ve funkci *diseqc* je pouze připraven tzv. *diseqc master command*, což je zpráva posílaná zařízením frontend do zařízení DiSEqC. Poté je tato zpráva předána funkci *diseqc_send_msg* ve formě struktury *dvb_diseqc_master_cmd*. Mimo tento parametr funkce přijímá další parametry důležité pro nastavení DiSEqC zařízení (*fe_sec_voltage_t, fe_sec_tone_mode_t, fe_sec_mini_cmd_t*). Ve funkci *diseqc_send_msg* je pouze nastaveno zařízení frontend podle přijatých parametrů pomocí funkce *ioctl*.

4.1.7 Získání dat

Pro získávání EPG informací z DVB vysílání bylo využito Linux DVB API, které je podrobněji popsané v sekci 3.2.1. Samotné získání pak provádí funkce *readData*, která je zobrazená ve výpisu 4. Tato funkce byla převzata a upravena od Ing. Davida Seidla, PhD.

```
int read_data(int defd, __u8 * data, int size_data , int pid){

    long dmx_buffer_size = TS_BUF_SIZE;

    if ( ioctl (defd,DMX_SET_BUFFER_SIZE,dmx_buffer_size) < 0){
        syslog(LOG_INFO, "set demux buffer failed");
```

```
        return 0;
    }

    struct dmx_sct_filter_params    sctFilterParams;
    memset(&sctFilterParams, 0, sizeof(struct dmx_sct_filter_params));
    sctFilterParams.pid=pid;
    sctFilterParams.timeout=3000;
    sctFilterParams.flags=DMX_IMMEDIATE_START|DMX_CHECK_CRC;

    if (  ioctl (defd,DMX_SET_FILTER,&sctFilterParams) < 0){
        syslog(LOG_INFO, "set demux filter failed");
        return 0;
    }

    int len = read(defd,data,size_data);

    return len;

}
```

Výpis 4: Funkce readData

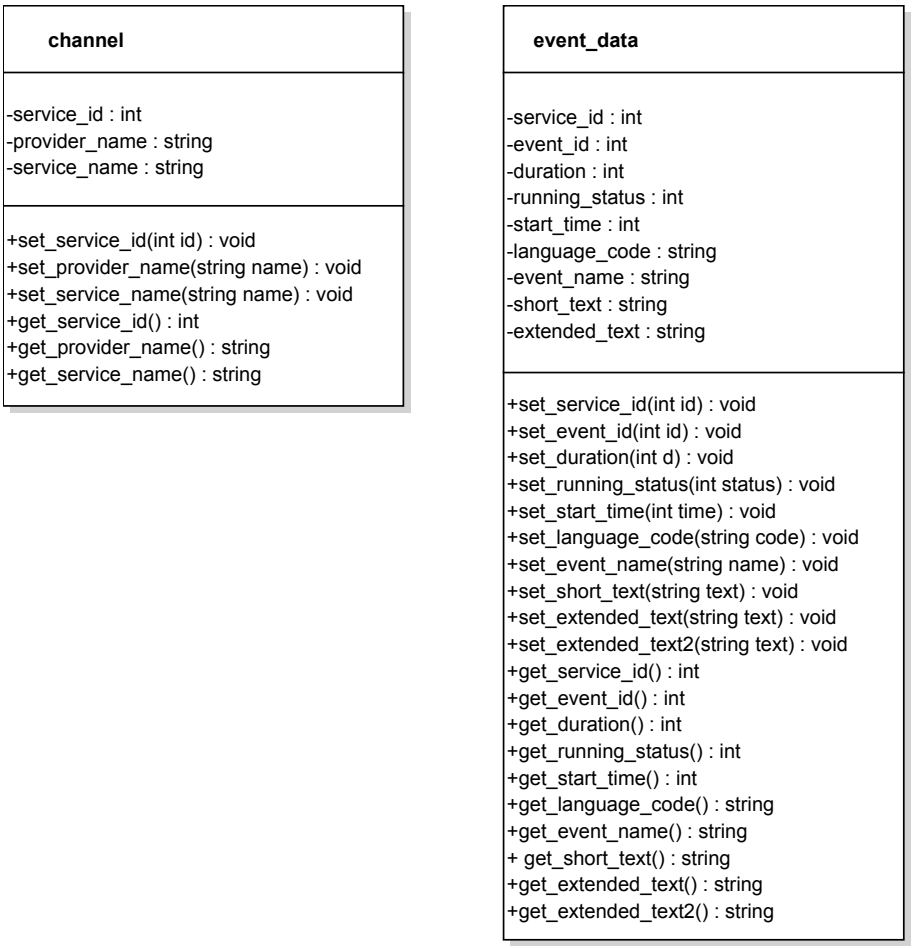
Funkce má čtyři parametry. Prvním parametrem je parametr *int defd*, který reprezentuje tzv. *File descriptor* získaný zavoláním funkce *open()*, která je součástí DVB API. Dalším parametrem je *__u8 * data*, který představuje buffer pro získaná data. Tento parametr souvisí i s parametrem následujícím, a to s parametrem *int size_data*, jenž přenáší velikost dat. Posledním parametrem funkce je parametr *int pid*. Tento parametr reprezentuje požadovaný PID.

Funkce nejprve nastaví velikost kruhové vyrovnávací paměti využívané pro filtrovaná data. K tomu dochází voláním funkce *ioctl* s parametry: file deskriptor *defd*, *DMX_SET_BUFFER_SIZE*, požadovaná velikost bufferu *dmx_buffer_size*. Dále je ve funkci vytvořen filtr, který umožňuje získat data s příslušným PID. Tento filtr se nastaví opět voláním funkce *ioctl* s parametry: file deskriptor *defd*, *DMX_SET_FILTER* a struktura *dmx_sct_filter_params* obsahující požadované parametry. Posledním krokem je samotné získání dat, k čemuž je využito metody z DVB API, konkrétně metody *read*.

Pro tuto práci jsou důležitá data v tabulkách s PID 17 a 18. PID 17 reprezentuje tabulku STD. Tato tabulka obsahuje data o programech (podrobnější popis proběhl v sekci 2.3.1). PID 18 reprezentuje tabulku EIT. Tato tabulka obsahuje data o jednotlivých pořadech (podrobnější popis proběhl v sekci 2.3.2).

4.1.8 Datové kontejnery

Pro uchování získaných dat bylo nutné navrhnout datovou strukturu. Díky zvolenému jazyku C++ bylo možné tyto kontejnery vytvořit pomocí tříd. Byly navrženy dvě třídy pro uchování informací o kanálech a programech. Pro informace o kanálech byla navržená třída *channelData* a pro informace o programech třída *eventData*. Strukturu těchto tříd zobrazuje třídní diagram na obrázku 12.



Obrázek 12: Třídní diagram datových kontejnerů

Každá z těchto dvou tříd obsahuje atributy a jejich getry a setry reprezentující jednotlivé EPG údaje, jako je například název programu atp.

Jelikož každá instance třídy tvoří pouze jeden pořad/program, tak bylo nutné tyto instance ukládat do kolekce. Jako vhodný typ datové kolekce byl zvolen *list*.

4.1.9 Zpracování dat

Zpracování dat se dá považovat za nejrozsáhlejší část aplikace. Data získaná pomocí funkce *read_data* jsou dále zpracovávána podle jejich typu. Pokud se jedná o informace o programech, jsou předány funkci *parse_channels* a pokud se jedná o informace o pořadech, jsou data předaná funkci *parse_events*. Zpracování dat v obou těchto případech je velice podobné, proto bude popsán pouze způsob zpracování dat, které obsahují informace o pořadech.

Jak již bylo zmíněno, data o pořadech získaná z vysílání s PID 18, jsou dále předány funkci *parse_events*. Kromě těchto dat funkce přijímá parametr reprezentující velikost těchto dat. V této funkci jsou data dekódována do struktury *section* pomocí metody *section_codec*, která je součástí knihovny *libucsi*. Poté je podle *table_id* rozhodnuto, zda-li se jedná o data o pořadech viz sekce 2.3.2 a tabulka 2. Následně proběhne další dekódování pomocí funkce *section_ext_decode*, tímto je získána struktura *section_ext*, která je nutná pro další dekódovací krok pomocí funkce *dvb_eit_section_codec*. Takto je získána struktura *dvb_eit_section*. Tato struktura obsahuje již data o určitých pořadech, které jsou uloženy v podobě pole struktur typu *dvb_eit_event*. Pro průchod tímto polem knihovna *libucsi* obsahuje speciální cyklus typu for-each (*dvb_eit_section_events_for_each*). Tímto cyklem jsou tedy procházeny jednotlivé pořady v podobě struktur *dvb_eit_event*, které obsahují pro tuto práci důležitá data. Tyto data jsou: event_id, start_time, duration, running_status, free_ca_mode a dále pole struktur deskriptorů. Jelikož nejzajímavější data jsou uloženy v deskriptorech, je nutné tyto deskriptory projít a data získat. K tomu účelu obsahuje knihovna *libucsi* opět speciální cyklus typu for-each (dvb_eit_event_descriptors_for_each). Tímto cyklem jsou tedy procházeny jednotlivé deskriptory, které jsou rozparsovávány pomocí funkce *parse_event_descriptor*. Tato funkce má dva parametry. První z nich je struktura reprezentující deskriptor, druhým parametrem je objekt typu *event_data* (viz sekce 4.1.8). Funkce *parse_events* je zobrazena ve výpisu 5.

```
void parse_events(uint8_t *buf, int len, list <event_data> &events)
{
    struct section *section;
    struct section_ext *section_ext = NULL;

    if ((section = section_codec(buf, len)) == NULL) {
        return;
    }

    switch(section->table_id) {

    case stag_dvb_event_information_nownext_actual:
    case stag_dvb_event_information_nownext_other:
```

```
case 0x50: case 0x51: case 0x52: case 0x53: case 0x54: case 0x55: case 0x56: case 0x57:
case 0x58: case 0x59: case 0x5a: case 0x5b: case 0x5c: case 0x5d: case 0x5e: case 0x5f:
case 0x60: case 0x61: case 0x62: case 0x63: case 0x64: case 0x65: case 0x66: case 0x67:
case 0x68: case 0x69: case 0x6a: case 0x6b: case 0x6c: case 0x6d: case 0x6e: case 0x6f:
{

    struct dvb_eit_section *eit;
    struct dvb_eit_event *cur_event;
    struct descriptor *curd;
    time_t start_time;

    if ((section_ext = section_ext_decode(section, 1)) == NULL) {
        return;
    }

    if (( eit = dvb_eit_section_codec(section_ext)) == NULL) {
        return;
    }

    dvb_eit_section_events_for_each(eit, cur_event) {

        event_data E = event_data();


        start_time = dvbdate_to_unixtime(cur_event->start_time);

        E.set_service_id(dvb_eit_section_service_id(eit));
        E.set_event_id(cur_event->event_id);
        E.set_duration(dvbduration_to_seconds(cur_event->duration));
        E.set_running_status(cur_event->running_status);
        E.set_start_time((int) start_time);

        dvb_eit_event_descriptors_for_each(cur_event, curd) {
            parse_event_descriptor(curd, E);
        }

        AddIntoEventList(events, E);

    }
    break;
}
}
}
```

Výpis 5: Funkce parseEvents

Ve funkci *parse_event_descriptor* je podle tagu deskriptoru rozhodnuto o jeho typu (viz tabulka 3 a podrobnější popis deskriptorů 2.4). V tomto případě jsou pro tuto práci důležité deskriptory nesoucí krátký a prodloužený popis pořadu (*short_event_descriptor, extended_event_descriptor*). Podle typu deskriptoru dále proběhne dekódování do struktury konkrétního deskriptoru a to opět pomocí funkcí z knihovny *libucsi*. V případě deskriptoru pro krátký popis pořadu se jedná o funkci *dvb_short_event_descriptor_codec* a strukturu *dvb_short_event_descriptor*. Nyní již máme přístup k datům uložených v deskriptoru.

V tomto případě jsou to data: event_name, short_text, language_code. Dekódování dat deskriptoru pro prodloužený popis pořadu je obdobný a získaná data jsou pouze extended_text. Funkce *parse_events_descriptor* zobrazena ve výpisu 6.

```
void parse_event_descriptor(struct descriptor *d,   event_data &E)
{

    switch(d->tag) {

    case dtag_dvb_short_event:
    {
        struct dvb_short_event_descriptor *dx;
        struct dvb_short_event_descriptor_part2 *part2;

        dx = dvb_short_event_descriptor_codec(d);
        if  (dx == NULL) {
            return;
        }
        part2 = dvb_short_event_descriptor_part2(dx);

        E.set_event_name(ToString(dx->event_name_length,
              dvb_short_event_descriptor_event_name(dx)));
        E.set_short_text(ToString(part2->text_length, dvb_short_event_descriptor_text(part2)));
        E.set_language_code(ToString(3, dx->language_code));
        break;
    }

    case dtag_dvb_extended_event:
    {
        struct dvb_extended_event_descriptor *dx;
        struct dvb_extended_event_descriptor_part2 *part2;

        dx = dvb_extended_event_descriptor_codec(d);
        if  (dx == NULL) {
            return;
        }

        part2 = dvb_extended_event_descriptor_part2(dx);

        E.set_extended_text(ToString(part2->text_length,
              dvb_extended_event_descriptor_part2_text(part2)));

        break;
    }

    }
}
```

Výpis 6: Funkce parseEventsDeskriptor

Jak již bylo zmíněno v části 2.5, jsou textová data v DVB kódována pomocí normy ISO 6937. Proto je nutné provézt jakousi konverzi. Pro tento převod je naimplementována funkce *ToString*. Jedná se pouze o dekódovací tabulku. Funkce má dva parametry.

První je délka tohoto textu a druhý je samotný text ve formátu ISO 6937. Ve funkci je tento text procházen po jednotlivých znacích. Pokud se jedná o ANSCI znak tzn. pokud je hexadecimální hodnota znaku menší než 0x7F, je tento znak přidán do výsledného řetězce. Pokud se jedná o hexadecimální hodnotu vyhrazenou pro diakritické znaménko (viz tabulka 4 v části 2.5), tak se přejde na další znak a podle typu diakritického znaménka a konstrukce switch se rozhodne o podobě výsledného znaku, který se přidá do výsledného řetězce. Výsledný řetězec je vrácen z funkce jako návratová hodnota typu string.

Takto získaná data jsou přiřazena do generických kolekcí typu list. První z kolekcí slouží pro uchování dat o programech, druhá slouží pro uchování dat o pořadech. Pro přidání dat do kolekcí jsou naimplementovány příslušné funkce, konkrétně tedy funkce *AddIntoEventList, AddIntoChannelList*. V těchto funkcích je procházen příslušný list prvků a je zjištěno, zda přidávaný prvek již v listu neexistuje, aby se zamezilo duplikaci dat. Rozhodnutí zda prvek již v kolekci existuje, je provedeno na základě porovnání příslušných *service_id* v případě programu a *event_id* v případě pořadu. Pokud prvek v kolekci neexistuje, tak je přidán.

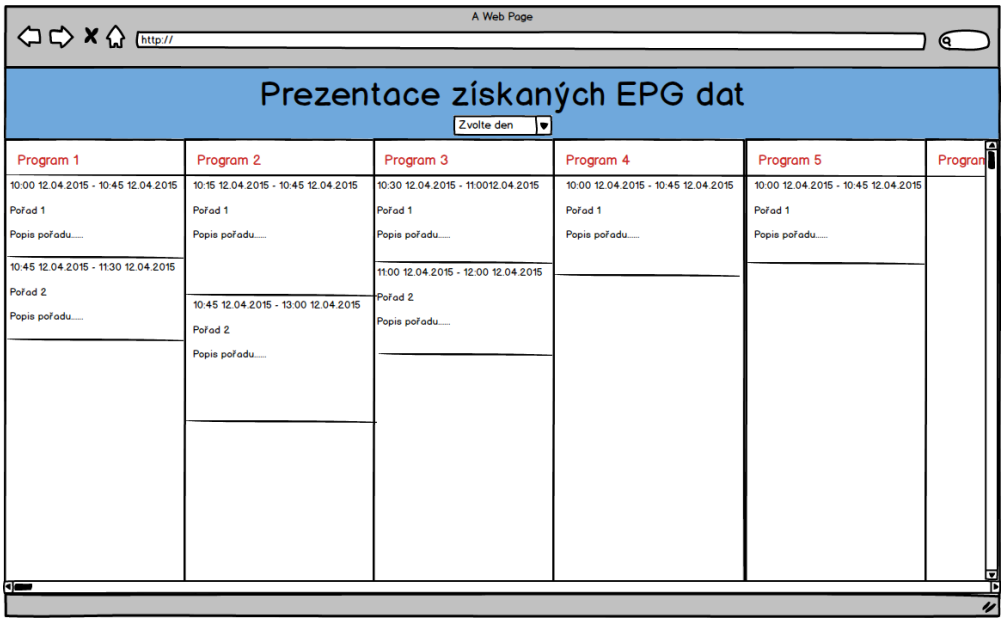
4.1.10 Export do XMLTV

Podle specifikace v zadání je jedním z požadavků ukládat získaná data ve formátu XMLTV tak, aby informace o programech byly uloženy v samostatných souborech. K tomuto účelu slouží funkce *ToXml*. Tato funkce má dva parametry. První reprezentuje list typu channel (list pro jednotlivé programy) a druhým je list typu event_data (list pro jednotlivé pořady). V této funkci je procházen v cyklu list pro programy. Podle tohoto listu se vytvářejí jednotlivé XML soubory. Tento cyklus obsahuje vnořený cyklus procházející jednotlivé pořady. Podle parametru service_id se rozhoduje, jestli pořad patří k určitému programu. V případě že ano, je tento pořad zapsán do XML souboru příslušného programu. Export do XML probíhá na konci každé iterace hlavní smyčky tzn. při dokončení procházení konfiguračního souboru.

4.2 Webové rozhraní

Jedním z požadavků kladených zadáním diplomové práce bylo vytvoření jednoduchého webového rozhraní pro prezentaci získaných EPG dat. Při vývoji byl kladen důraz na jednoduchost a přehlednost. Při navrhování aplikace bylo vycházeno z již existujících aplikací a zaběhlých zvyklostí. Příkladem můžou být obyčejné televizní noviny nebo televizní program poskytovaný službou *seznam.cz* dostupný na adrese `http://tv.seznam.cz/`. Návrh webového rozhraní je zobrazen na obrázku 14.

Pro vývoj bylo využito jazyků HTML, JavaScript, PHP, CSS. Při vývoji bylo postupováno podle návrhu. Webové rozhraní uspořádává data jednotlivých programů do sloupců. Pořady jednotlivých programů jsou setříděny od nejstarších po nejnovější. Do rozhraní byla naimplementována možnost zobrazit pouze pořady pro určitý den.



Obrázek 13: Návrh webového rozhraní

4.2.1 Popis funkcionality

Pomocí JavaScriptu respektive JQuery jsou naimplementovány dvě funkce, a to konkrétně funkce *getData* a funkce *getList*. Úkolem funkce *getList* je naplnit dropdown list možnými dny pro výběr EPG dat, které získá pomocí PHP skriptu. Komunikace je provedena pomocí metody *get*. Poté funkce čeká na odpověď od serveru s daty strukturovanými pro html dropdownlist. Tato data jsou pouze přidána jako možnost výběru do dropdown listu.

Úkolem funkce *getData* je předat PHP skriptu hodnotu z dropdown listu, která reprezentuje požadovaný den a vyčkat na požadovaná data. Výchozí hodnota zajistí získání dat všech. Komunikace je provedená pomocí metody *post*. Data jsou jako v předchozím případě již připravena do požadované struktury a stačí je pouze přidat na stránku.

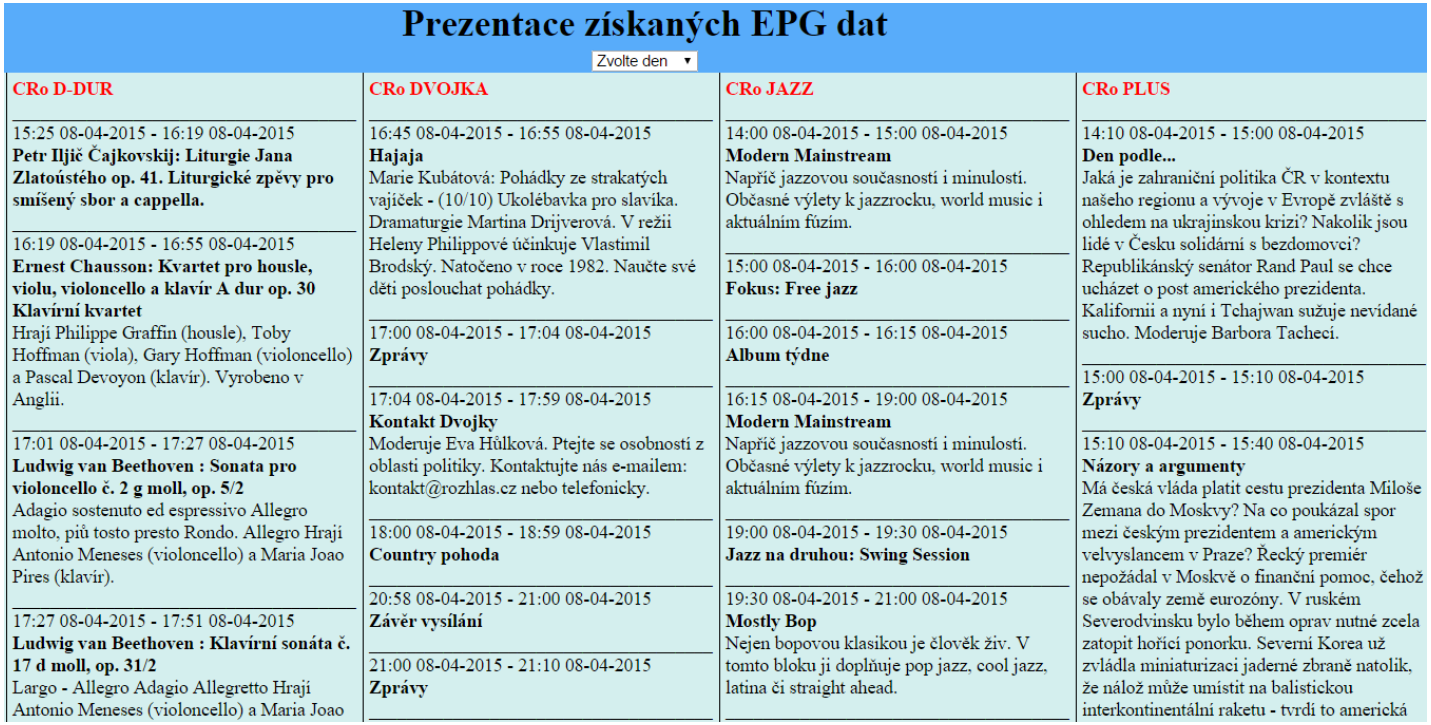
Jak již bylo zmíněno, serverová strana připravuje požadovaná data pomocí dvou PHP skriptů. Skript pro generování dat pro dropdown list a skript pro přípravu EPG dat. Skript generující data pro dropdown list projde všechny XML soubory, reprezentující EPG data, v adresáři pomocí cyklu foreach. Vnořený foreach cyklus prochází XML soubor a vybírá pouze datum začátku pořadu, které následně ukládá do pole (php metodou in_array je ověřováno, jestli již v poli tento datum není). Po projití všech XML souborů v adresáři by v poli měla být uložena všechna data. Toto pole je pak setříděno a následně projeto foreach cyklem a jednotlivá data jsou strukturována do podoby HTML tagu *option*, který reprezentuje jednotlivé volby v dropdown listu. Druhým skriptem je skript

pro získání EPG dat. Tomuto skriptu jsou předána data reprezentující požadovaný den. Ve skriptu jsou dále procházeny XML soubory s EPG daty. Z XML souboru je nejprve získán název programu, který je hned převeden do požadované podoby a vrácen funkcí *echo*. Dále jsou vnořeným cyklem procházeny jednotlivé informace o pořadech, které jsou také ihned strukturovány do požadované podoby a vráceny funkcí *echo*. Tímto způsobem jsou projeté veškeré XML soubory s EPG daty. Tato funkce je zobrazena ve výpisu 7.

```
<?php

  $day = $_POST["den"];

    $files  = glob("EPG/*");
    foreach( $files  as  $file )  {
        $xml=simplexml_load_file($file);
        echo "<td>";
        echo "<span␣class=\"nazev\">" . $xml->channel->{'display-name'} . "</span></br>";
        echo "_____</br>";

        foreach($xml->programme as $program) {
          $start  = $program['start' ];
          $stop = $program['stop'];

          $dt = new DateTime("@$start");

      if ($day == $dt->format('d-m-Y') || $day == "all"){
          echo $dt->format('H:i␣d-m-Y') . "␣-␣";
          $dt = new DateTime("@$stop");
          echo $dt->format('H:i␣d-m-Y') . "</br>";
          echo "<b>" .  $program->title . "</b></br>";
          if ( strlen ($program->desc) > 0 ){
           echo $program->desc . "</br>";
          }
          echo "_____</br>";

          }
        }
        echo "</td>";
    }
?>
```

Výpis 7: PHP skript pro předání EPG dat

Výsledná webová aplikace pro reprezentaci získaných EPG dat je zobrazena na obrázku 14. Z obrázku je zřetelná sloupcová struktura zmíněna na začátku této podkapitoly. Dále je možné vidět ovládací prvek pro výběr EPG dat pro určitý den.



Obrázek 14: Výsledná webová aplikace

4.3 Testování

Testování probíhalo výhradně na systému DVB-T. To bylo umožněno díky podobnosti se systémem DVB-S. Odlišnosti jednotlivých systémů zasáhly pouze do několika funkcí. Jednou z těchto funkcí byla funkce pro ladění *tune*, kde je rozdíl v parametrech potřebných pro ladění jednotlivých kanálů. Díky odlišnosti těchto parametrů je jasné, že další upravenou funkcí musí být funkce pro rozparsovávání konfiguračních souborů, konkrétně tedy callback funkce *channels_cb*.

Bylo objeveno a opraveno několik drobných chyb jak v aplikaci pro získávání EPG dat z DVB vysílání, tak i u aplikace pro prezentování výsledných EPG dat. Jednou z nalezených chyb u aplikace pro získávání EPG dat bylo špatné přiřazování dat do proměnných objektů reprezentujících jednotlivé pořady. Díky této chybě byla do datového kontejneru, v případě dlouhých popisů jednotlivých pořadů, uložena pouze poslední část dat (z posledního deskriptoru *extended_event_descriptor*). Jak už bylo zmíněno v části 2.4.3, tak v případě prodloužených popisů pořadu přenášených pomocí deskriptorů *extended_event_descriptor* může být popis pořadu přenášen v několika těchto deskriptorech. Výsledný text je poté nutné složit ze všech přijatých deskriptorů pro tento pořad. Pro opravu této chyby stačilo změnit operátor přiřazení (=) ve funkci *setExtendedText* za kombinaci operátoru přiřazení s operátorem plus (+=). Touto malou změnou bylo dosaženo požadované funkcionality. Příkladem nalezené chyby ve webové aplikaci pro prezentaci získaných EPG dat může být chybné zobrazování některých speciálních znaků v názvech pořadů. Jedním z těchto znaků je znak *&*, který v HTML má speciální funkci. Možností řešení této chyby bylo více, ale jako nejefektivnější bylo zvoleno upravení funkce *ToString* v aplikaci pro získávání EPG dat. Jak už bylo zmíněno v části 4.1.9, ve funkci je proveden převod textu z formátu ISO 6937, přičemž je procházený text po jednotlivých znacích. Do funkce byla přidaná podmínka pro dekódování znaku *&*, jako *&* což je ekvivalentní kódové označení znaku *&* pro HTML.

Jelikož získávání EPG dat je velice náročné na čas, tak byla provedena optimalizace pomocí vláken, kdy po naladění na určitý kanál bylo zahájeno získávání dat ve čtyřech vláknech. Teoreticky by tak rychlost získávání dat měla být čtyřnásobná. Prakticky se však jedná o mnohem menší zrychlení, jelikož vlákna získají z velké části stejná data. Do vláken bylo přidáno zpoždění, aby sběr dat ve vláknech startoval v různých okamžicích. Tímto se podařilo dosáhnout lepší optimalizace.

4.4 Možnosti dalšího rozšíření a vylepšení

Aplikace dokáže získávat EPG data ze zařízení DVB-S a dále také DVB-T. Jedním z možných rozšíření by mohla být podpora dalších zařízení, například DVB-C nebo v USA používaného ATSC.

Aplikace vyžaduje pro svou funkci konfigurační soubor s jednotlivými kanály. Tento soubor je nutné vygenerovat před spuštěním aplikace softwarem třetí strany (viz 4.1.2). Dalším rozšířením aplikace by tedy mohla být možnost automatického generování konfiguračního souboru v rámci aplikace.

```
NOVA:602000000:INVERSION_AUTO:BANDWIDTH_8_MHZ:FEC_2_3:FEC_AUTO:QAM_64:TRANSMISSION_MODE_8K:GUARD_INTERVAL_1_4:HIERARCHY_NONE:101:111:513
NOVA CINEMA:602000000:INVERSION_AUTO:BANDWIDTH_8_MHZ:FEC_2_3:FEC_AUTO:QAM_64:TRANSMISSION_MODE_8K:GUARD_INTERVAL_1_4:HIERARCHY_NONE:401:411:514
Prima COOL:602000000:INVERSION_AUTO:BANDWIDTH_8_MHZ:FEC_2_3:FEC_AUTO:QAM_64:TRANSMISSION_MODE_8K:GUARD_INTERVAL_1_4:HIERARCHY_NONE:501:511:770
Prima:602000000:INVERSION_AUTO:BANDWIDTH_8_MHZ:FEC_2_3:FEC_AUTO:QAM_64:TRANSMISSION_MODE_8K:GUARD_INTERVAL_1_4:HIERARCHY_NONE:601:611:773
BARRANDOV TV:602000000:INVERSION_AUTO:BANDWIDTH_8_MHZ:FEC_2_3:FEC_AUTO:QAM_64:TRANSMISSION_MODE_8K:GUARD_INTERVAL_1_4:HIERARCHY_NONE:301:311:2050
CT D / CT art:610000000:INVERSION_AUTO:BANDWIDTH_8_MHZ:FEC_2_3:FEC_AUTO:QAM_64:TRANSMISSION_MODE_8K:GUARD_INTERVAL_1_4:HIERARCHY_NONE:2031:2032:264
RETRO MUSIC TV:610000000:INVERSION_AUTO:BANDWIDTH_8_MHZ:FEC_2_3:FEC_AUTO:QAM_64:TRANSMISSION_MODE_8K:GUARD_INTERVAL_1_4:HIERARCHY_NONE:2101:2102:5889
CT 1 HD:610000000:INVERSION_AUTO:BANDWIDTH_8_MHZ:FEC_2_3:FEC_AUTO:QAM_64:TRANSMISSION_MODE_8K:GUARD_INTERVAL_1_4:HIERARCHY_NONE:2001:2002:261
CT 2 HD:610000000:INVERSION_AUTO:BANDWIDTH_8_MHZ:FEC_2_3:FEC_AUTO:QAM_64:TRANSMISSION_MODE_8K:GUARD_INTERVAL_1_4:HIERARCHY_NONE:2011:2012:263
CT sport HD:610000000:INVERSION_AUTO:BANDWIDTH_8_MHZ:FEC_2_3:FEC_AUTO:QAM_64:TRANSMISSION_MODE_8K:GUARD_INTERVAL_1_4:HIERARCHY_NONE:2021:2022:262
RADIO CAS ROCK:610000000:INVERSION_AUTO:BANDWIDTH_8_MHZ:FEC_2_3:FEC_AUTO:QAM_64:TRANSMISSION_MODE_8K:GUARD_INTERVAL_1_4:HIERARCHY_NONE:0:2212:17922
KINOSVET:610000000:INVERSION_AUTO:BANDWIDTH_8_MHZ:FEC_2_3:FEC_AUTO:QAM_64:TRANSMISSION_MODE_8K:GUARD_INTERVAL_1_4:HIERARCHY_NONE:2301:2302:6145
RADIO CAS:610000000:INVERSION_AUTO:BANDWIDTH_8_MHZ:FEC_2_3:FEC_AUTO:QAM_64:TRANSMISSION_MODE_8K:GUARD_INTERVAL_1_4:HIERARCHY_NONE:0:2202:17921
CT  D / CT art:690000000:INVERSION_AUTO:BANDWIDTH_8_MHZ:FEC_2_3:FEC_AUTO:QAM_64:TRANSMISSION_MODE_8K:GUARD_INTERVAL_1_4:HIERARCHY_NONE:6145:6146:264
```

Obrázek 15: Ukázka konfiguračního souboru

V případě nepřetržitého chodu aplikace by bylo dobré zvažovat vylepšení správy paměti. Aplikace ukládá data do kolekce, která se stále rozšiřuje, čímž rostou nároky na operační paměť. Vylepšení by mohlo být provedeno například tak, že by se provádělo rozšiřování XML souboru novými EPG daty namísto rozšiřování kolekce a kompletního přepisu XML souboru. Dalším rozšířením aplikace souvisejícím se správou paměti, by mohla být implementace skriptu, který by byl spouštěn v nějakých časových intervalech a jehož úkolem by bylo mazání starých EPG dat z XML souboru.

4.5 Použití aplikace

Aplikace je přiložena ve formě zdrojových kódů, proto před použitím musí být nejdříve přeložena. Pro usnadnění je přiložen *Makefile*. Další podmínkou je existence konfiguračního souboru.

4.5.1 Potřebné knihovny a jejich instalace

Pro správnou funkčnost je nutné mít v počítači nainstalováno *Linux DVB API* a knihovnu *libucsi* (viz. sekce 3.2). Jelikož všechny potřebné knihovny obsahuje balík *dvb-apps*, je nejjednodušším řešením tedy instalace tohoto balíku. Instalaci lze provézt v distribuci *Ubuntu* příkazem:

```
sudo apt-get install dvb-apps
```

Tímto získáme vše potřebné pro přeložení aplikace. Pro samotné přeložení je přiložen *Makefile* a je nutné se pouze přesunout do adresáře aplikace, například pomocí příkazu *cd* a spustit přeložení příkazem *make*.

4.5.2 Konfigurační soubor

Pro správnou funkci aplikace je nutné mít konfigurační soubor. Jak tento soubor získat, bylo popsáno v sekci 4.1.2. Struktura souboru je zobrazena na obrázku 15.

4.5.3 Spuštění aplikace

Aplikaci je možné spustit z adresáře aplikace pomocí příkazu *./epgread*. Byla také naimplemetována možnost upřesnit některá nastavení aplikace pomocí parametrů. Jednotlivé parametry a jejich popis je zobrazen v následujícím výpisu:

- **-a číslo:** číslo použitého adaptéru (výchozí *0*),
- **-f číslo:** číslo použitého frontend zařízení (výchozí *0*),
- **-d číslo:** číslo použitého demux zařízení (výchozí *0*),
- **-c soubor:** cesta ke konfiguračnímu souboru (výchozí */tmp/channels.conf*),
- **-l číslo:** číslo reprezentující typ LNB (výchozí *0*):

  0 - UNIVERSAL,
  1 - DBS,
  2 - STANDARD,
  3 - ENHANCED,
  4 - C-BAND,

- **-t cesta:** cesta k XML souborům (výchozí */tmp/XML/*).

Neúspěšné spuštění aplikace se zaznamenává v systémovém logu, který se nachází v */var/log/syslog*.

4.5.4 Webové rozhraní

Aplikace pro reprezentaci EPG dat vyžaduje webový server, například Apache s nainstalovaným PHP. Aplikace zobrazuje obsah všech XML souborů s EPG daty ve složce XML, kterou je nutné spolu se zdrojovými kódy webového rozhraní nakopírovat na server. Přesným postupem instalace potřebného softwaru se zabývá příloha B.2.

5 Závěr

Cílem práce bylo navrhnout a realizovat systém, který bude prostřednictvím DVB-S/S2 karty získávat EPG informace vysílané na jednotlivých transpondérech družice. Dále vytvořit jednoduchý webový interface pro reprezentaci EPG dat a jednotlivé aplikace otestovat.

V rámci práce byly jednotlivé cíle splněny. Výsledná aplikace navíc umožňuje získávat EPG data také ze systému DVB-T. Webový interface umožňuje zobrazovat získaná data roztříděná dle programu a seřazená podle času. Navíc je naimplementována možnost vybírat program jen pro určitý den. V rámci testování bylo objeveno několik drobných chyb, jenž byly popsány v kapitole zabývající se testováním 4.3, dále bylo testování zaměřeno na zvýšení rychlosti získávání EPG dat. Toho bylo docíleno pomocí zpracování ve vláknech. Získávání dat bylo tedy rozděleno do čtyřech vláken, ale jelikož vlákna získávala z velké části stejná data, zdálo se rozdělení do vláken zbytečné. Tento problém se podařilo částečně vyřešit prodlevou před spuštěním každého vlákna.

První kapitola práce pojednává o digitálním vysílání a jeho zpracování. Je zde tedy krátký historický náhled na vývoj DVB, dále základní popis technologie, který je následován multiplexováním. Kapitola dále pokračuje popisem vybraných servisních informačních tabulek a deskriptorů. Poslední část kapitoly je věnována EPG datům.

Druhá kapitola popisuje jednotlivé možnosti řešení, jako je volba programovacího jazyka. Dále jsou v této kapitole zmíněny vhodné knihovny pro tuto práci, konkrétně tedy Linux DVB API a balík knihoven dvb-apps.

Třetí kapitola se věnuje samotnému řešení. V první části je popsán vývoj a funkcionalita aplikace pro získávání EPG dat z DVB-S/S2 vysílání. Druhá část je věnována webovému rozhraní, kde je opět popsán vývoj a funkcionalita aplikace. Třetí část této kapitoly je zaměřena na testování. Čtvrtá část kapitoly obsahuje možnosti dalšího rozšíření a vylepšení. V poslední části je popsáno použití aplikace.

6 Reference

[1] CHERRYHILL.EU. *Cherry EPG: What is EPG?* [online]. 2012 [cit. 2015-05-03]. Dostupné z: http://epg.cherryhill.eu/what

[2] DVB PROJECT. *DVB: History of DVB* [online]. 2013 [cit. 2015-05-03]. Dostupné z: https://www.dvb.org/about/history

[3] EN 300 468. *Digital Video Broadcasting (DVB); Specification for Service Information (SI) in DVB systems*. Valbonne: European Telecommunications Standards Institute, 1998. Dostupné z: http://www.etsi.org/deliver/etsi_en/300400_300499/300468/01.03.01_60/en_300468v010301p.pdf

[4] LEGÍŇ, Martin. *Televizní technika DVB-T*.1. vyd. Praha: BEN - technická literatura, 2006, 286 s. ISBN 80-730-0204-3.

[5] METZLER, Ralph J.K. a Marcus O.C. METZLER. *LINUX DVB API* [online]. 2003 [cit. 2015-05-03]. Dostupné z: http://www.linuxtv.org/docs/dvbapi/dvbapi.html

[6] LINUXTV. *LinuxTV dvb-apps* [online]. 2013 [cit. 2015-05-03]. Dostupné z: http://www.linuxtv.org/wiki/index.php/LinuxTV_dvb-apps

[7] TOMAN, Jiří a Ivo PROCHÁZKA. ČESKÁ TELEVIZE. *Technické základy DVB-T* [online]. [cit. 2015-05-03]. Dostupné z: http://www.ceskatelevize.cz/vse-o-ct/technika/digitalni-pozemni-vysilani-dvb-t/technicke-zaklady/

[8] XMLTV.ORG. *XMLTV File format* [online]. 2008 [cit. 2015-05-03]. Dostupné z: http://wiki.xmltv.org/index.php/XMLTVFormat

[9] WIKIPEDIA.ORG. *DVB-T* [online]. 2015 [cit. 2015-05-03]. Dostupné z: http://en.wikipedia.org/wiki/DVB-T

[10] THE LINUX INFORMATION PROJECT. *Daemon Definition* [online]. 2005 [cit. 2015-05-03]. Dostupné z: http://www.linfo.org/daemon.html

[11] WATSON, Devin. *Linux Daemon Writing HOWTO* [online]. 2004 [cit. 2015-05-03]. Dostupné z: http://www.netzmafia.de/skripten/unix/linux-daemon-howto.html

[12] JDS UNIPHASE CORPORATION. *MPEG-2 and DVB digital broadcasting* [online]. 2006 [cit. 2015-05-03]. Dostupné z: http://www.jdsu.com/ProductLiterature/MPEG_Poster_lowrez.pdf

A Tabulky

nibble_level_1	nibble_level_2	Description
0x0	0x0 to 0xF	undefined content
		Movie/Drama:
0x1	0x0	movie/drama (general)
	0x1	detective/thriller
	0x2	adventure/western/war
	0x3	science fiction/fantasy/horror
	0x4	comedy
	0x5	soap/melodrama/folkloric
	0x6	romance
	0x7	serious/classical/religious/historical movie/drama
	0x8	adult movie/drama
	0x9 to 0xE	reserved for future use
	0xF	user defined
		News/Current affairs:
0x2	0x0	news/current affairs (general)
	0x1 n	ews/weather report
	0x2	news & magazine
	0x3	documentary
	0x4	discussion/interview/debate
	0x5 to 0xE	reserved for future use
	0xF	user defined
		Show/Game show:
0x3	0x0	show/game show (general)
	0x1	game show/quiz/contest
	0x2	variety show
	0x3	talk show
	0x4 to 0xE	reserved for future use
	0xF u	ser defined
		Sports:
0x4	0x0	sports (general)
	0x1	special events (Olympic Games, World Cup, etc.)
	0x2	sports magazines
	0x3	football/soccer
	0x4	tennis/squash
	0x5	team sports (excluding football)
	0x6	athletics
	0x7	motor sport
	0x8	water sport
	0x9	winter sports
	0xA	equestrian

	0xB	martial sports
	0xC to 0xE	reserved for future use
	0xF	user defined
		Children's/Youth programmes:
0x5	0x0	children's/youth programmes (general)
	0x1	pre-school children's programmes
	0x2	entertainment programmes for 6 to14
	0x3	entertainment programmes for 10 to 16
	0x4	informational/educational/school programmes
	0x5	cartoons/puppets
	0x6 to 0xE	reserved for future use
	0xF	user defined
		Music/Ballet/Dance:
0x6	0x0	music/ballet/dance (general)
	0x1	rock/pop
	0x2	serious music/classical music
	0x3	folk/traditional music
	0x4	jazz
	0x5	musical/opera
	0x6	ballet
	0x7 to 0xE	reserved for future use
	0xF	user defined
		Arts/Culture (without music):
0x7	0x0	arts/culture (without music, general)
	0x1	performing arts
	0x2	fine arts
	0x3	religion
	0x4	popular culture/traditional arts
	0x5	literature
	0x6	film/cinema
	0x7	experimental film/video
	0x8	broadcasting/press
	0x9	new media
	0xA	arts/culture magazines
	0xB	fashion
	0xC to 0xE	reserved for future use
	0xF	user defined
		Social/Political issues/Economics:
0x8	0x0	social/political issues/economics (general)
	0x1	magazines/reports/documentary
	0x2	economics/social advisory
	0x3	remarkable people
	0x4 to 0xE	reserved for future use

	0xF	user defined
		Education/Science/Factual topics:
0x9	0x0	education/science/factual topics (general)
	0x1	nature/animals/environment
	0x2	technology/natural sciences
	0x3	medicine/physiology/psychology
	0x4	foreign countries/expeditions
	0x5	social/spiritual sciences
	0x6	further education
	0x7	languages
	0x8 to 0xE	reserved for future use
	0xF	user defined
		Leisure hobbies:
0xA	0x0	leisure hobbies (general)
	0x1	tourism/travel
	0x2	handicraft
	0x3	motoring
	0x4	fitness and health
	0x5	cooking
	0x6	advertisement/shopping
	0x7 g	ardening
	0x8 to 0xE	reserved for future use
	0xF	user defined
		Special characteristics:
0xB	0x0	original language
	0x1	black and white
	0x2	unpublished
	0x3	live broadcast
	0x4	plano-stereoscopic
	0x5	local or regional
	0x6 to 0xE	reserved for future use
	0xF	user defined
		Reserved for future use:
0xC to 0xE	0x0 to 0xF	reserved for future use
		User defined:
0xF	0x0 to 0xF	user defined

Tabulka 5: Tabulka pro kódování content deskriptoru [3]

B Implementace

Součástí práce je také CD s implementací a elektronickou podobou práce ve formátu PDF.

B.1 Instalace aplikace pro získávání EPG dat

1. Instalace potřebných knihoven - *sudo apt-get install dvb-apps*,
2. přeložení zdrojových kódů - *make*,
3. vygenerování konfiguračního souboru:

DVB-S: *scan /usr/share/dvb/dvb-s/Astra-23.5E > /tmp/channels.conf*,
DVB-T: *scan /usr/share/dvb/dvb-t/cz-All > /tmp/channels.conf*,

4. spuštění aplikace s výchozím nastavením (viz. sekce 4.5.3) - *./epgread*.

B.2 Instalace webového serveru

1. Instalace webového serveru Apache:
   - *sudo apt-get update*,
   - *sudo apt-get install apache2*,
2. instalace PHP - *sudo apt-get install php5 libapache2-mod-php5 php5-mcrypt*,
3. dalším krokem je nastavení práv pro přístup k souborům v adresáři *var/www*:
   - *sudo chgrp -R www-data /var/www*,
   - *sudo usermod -a -G www-data demousername,*
   - *sudo chmod -R 2770 /var/www*,
4. dále je nutné nakopírování zdrojových souborů, včetně vygenerovaných EPG dat ve formátu XML do adresáře */var/www/*,
5. posledním krokem je restartování webového serveru příkazem - *sudo service apache2 restart*.

Pokud je vše správně nastavené a nainstalované, přístupem na `http://localhost`, je možné spustit webové rozhraní pro prezentaci EPG dat.